# **Overview**

DevExtreme is a set of enterprise-ready UI component suites for Angular, React, Vue, and jQuery. It is everything you need to create responsive web apps for touch devices and traditional desktops: data grid, interactive charts, data editors, navigation and multi-purpose widgets. These controls are designed to look great and to provide powerful functionality in any browser.

DevExtreme provides ready to use templates and demos for **Data grid, Pivot Grid, Tree List, Scheduler, HTML Editor, Diagram, Charts, Gantt, Gauges, Navigation, Editors, Forms and Multi-Purpose, File Management, Actions and List, Maps, Dialogs and Notification.**

DevEtreme Provide collection of UI elements like **Accordion, ActionSheet, Autocomplete, BarGauge, Box, Bullet, Button, ButtonGroup, Calendar, Chart, CheckBox, CircularGauge, ColorBox, ContextMenu, DataGrid, DateBox, Diagram, Drawer, DropDownBox, DropDownButton, FileManager, FileUploader, FilterBuilder, Floating Action Button, Form, Funnel, Gallery, Gantt, HtmlEditor, LinearGauge, List, LoadIndicator, LoadPanel, Lookup, Map, Menu, MultiView, NumberBox, PieChart, PivotGrid, PolarChart, Popover, Popup, ProgressBar, RadioGroup, RangeSelector, RangeSlider, Resizable, ResponsiveBox, Sankey, Scheduler, ScrollView, SelectBox, Slider, Sparkline, Switch ,TabPanel, Tabs ,TagBox ,TextArea , TextBox, TileView, Toast, Toolbar ,Tooltip ,TreeList ,TreeView, VectorMap.**

# **DevExtrem with Jquery**

# **DevExtreme jQuery Components**

The DevExtreme jQuery Component Suite is a feature-complete set of 65+ responsive and touch-enabled UI components implemented as jQuery plugins. The components are a data grid, interactive charts, data editors, navigation and multi-purpose UI components.

In addition to components, the suite includes the following features:

* [Data layer](https://js.devexpress.com/Documentation/20_2/Guide/Data_Binding/Data_Layer/)
* [Client-side data validation](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/Common/UI_Widgets/Data_Validation/)
* [Themes and styles](https://js.devexpress.com/Documentation/20_2/Guide/Themes_and_Styles/Predefined_Themes/)
* [Localization](https://js.devexpress.com/Documentation/20_2/Guide/Common/Localization/)
* [Modularity](https://js.devexpress.com/Documentation/20_2/Guide/Common/Modularity/Link_Modules/)
* [Customization using templates](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/Common/Templates/)

These CDN links needed to use DevExtreme Library in Our Jquery Application.

* <script type="text/javascript" src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
* <link rel="stylesheet" href="https://cdn3.devexpress.com/jslib/20.2.10/css/dx.common.css">
* <link rel="stylesheet" href="https://cdn3.devexpress.com/jslib/20.2.10/css/dx.light.css">
* <script type="text/javascript" src="https://cdn3.devexpress.com/jslib/20.2.10/js/dx.all.js"></script>

# **Component Configuration Syntax**

Every UI Component must be within container call <div> and must have a unique ID.

Then later this container will be used for plugin DevExtreme UI components via JQuery.

Later we can configure that component by passing an argument to that plugin.

## **Get and Set Properties**

For getting Property we can create an instance of UI Component bypassing “instance” and access get all the properties of that instance via its option method or get single property via passing the name of the property to that option method.

We can also directly get all the properties or single properties without crating instances bypassing the ‘option’ parameter to that components.

We can set property value via the same get method with an extra parameter that contains a value for that property.

We can set multiple properties via passing a single object literal which contains a key as the property name and value as the value for that property.

We can do that with or without creating an instance of that component.

## **Call Methods**

We can call the components method via the same way we get and set property.

We just need to pass the name of the method and its argument to the plugin or by creating its instance and we can just call that method.

By creating an instance we can just call the method we don’t need to pass the name of that method as an argument to that instance.

## **Handle Events**

We can subscribe to the event in the same way to set property.

All we need to just pass an object literal to that plugin that contains the key as event name and value as a function that handles that event.

We can pass multiple events to one object literal.

We can also call the .on method of instance to set an event also.

To unsubscribe that event we can call the .off method which contains the event name and method which we want to reattach.

## **Dispose of a UI Component**

For Disposing UI component, we can pass ‘dispose’ as a parameter and then remove its container via calling the remove() method.

# **Editor (Overview)**

## **CheckBox**

This element is used when user want select or deselect something on display.

These are importans properties of CheckBox.

accessKey :- this property is used to access this element when user press key on keyboard.

Disabled:- this property is used to enable disable checkbox element.

elementAttr:- this property is used to set value to to current element’s attribute. We can pass object which contains all the value for this element.

Height:- this property is used to set height of checkbox.

Hint:- this property is used to display hint for checkbox.

isValid:- this property is used to check that checkbox value passed all the validation rule or not.

Name: this property is used to set name of checkbox container.

readOnly:- This property is used to set whether this checkbox is readonly or not.

Text:- This property is used to set lable for current checkbox.

validationError:- this property is used to check all the validation errors.

validationMessageMode:- this property is used to set how error message will be displayed. Value could be auto or always.

value:- either true, false or undefined,

visible:- this property indicate that whether checkbox should be visible or not.

Width:- this property used to set width of checkbox.

Theses are importants methods of checkbox:

defualtOption(rule): this method accept object which contains device and options attribute to set particular options for specific type of device.

Dispose() This method is used to dispose checkbox instance.

Element() this method is used to get HTMLEelement of checkbox.

Focus() this method is used to focus on checkbox.

Getinstanse() this method is used to getinstance of HTML element.

Instance() this method is used to get instance current instance.

Off(eventname) this method is used to deattach event handler from particular event.

On(eventname) this method is used to attach event handle for specific event.

Option(name,value) this method is used to set option for checkbox.

registerHandler(key,method): this method is used to register specific keyboard key to element.

These are usefull events that used in checkbox.

contentReady: Raised when element is ready.

Disposing: Raised when element is disposing.

Initialized: Raised when element is initialized.

Optionchanged: Raised when element optionchanged.

valueChanged: Raised when element ValueChanged.

## **DateBox**

Date box is used to get Date or Time or Date and Time from user.

It will show user a Datebox where user can choose Date or time which he/she wants to enter.

These are the usefull properties of DateBox:

acceptCustomValue: Accept boolen value that indicate that whether user can add custome date or not by typing.

adaptivityEnabled: This property is used to set adaptivity for different type of devices and screens. It will indicate that whether clock is displayed or not based on screen and device.

applyButtonText: Text for Ok Button.

applyValueMode: Indicate that value will be instantly selectected or user have to use ok button.

Buttons[]: it is used to display button withing DateBox. Only accepted button name as string here is clear and dropdown. We can also provide custome buton by providing object of it.

Button objcect contains location, name, and option property.

Where option contains button’s configurationProperty like , stylingMode, width, elementAttr, onClick.

calendarOptions: These option is useful when pickertye is calender and we can configure calander property by using this property.

dateOutOfRangeMessage: this property is usefull for display message when date is out of Ranged.

Default max is 50 years from current date and defult min is 1900 year.

dateSerializationFormat: is used to serilized date formate vlkue.

disabledDates: this property usefult when pickertype is calander. This property accepts array of Dates which we want to disable on calnder so user cannot pickeup dates.

displayFormate: This property is set formate of date and time in UI element that displays to user.

inputAttr: This property is used to set attributes value of input element of this Datebox.

invalidDateMessage: This property is usefult to set Error message when user enters invalid date and time formate.

Interval: This property is used when type is time and pickertype is list. This property is used to set interval between Time.

Max: This property is used to set max date user can enter. Defualt value is 50 years from current date.

Min: This property is used to set minimum of date user can enter. Default value is 1900 year.

pickerType: This is very useful property. This property is used to set Date picker type. User can set native, calender, roller or List Property here.

Placeholder: This property is used set placeholder of dateBox.

showAnalogClock: This property is used to set whether analogclock should displayed or not. This prorty is only useful when pickertye is calender.

showClearButton: This property is used to set whether Clear Button shold displayed or not.

showDropDownButton: This property is used to set whether Dropdown Button shold displayed or not.

Spellcheck: This property is used to set whether UI displays speling mistake or not.

stylingMode: This property is used to stylingmode for Datebox. This could be outlined, underlined and filled.

Text: This property is used to display Text value of Datebox. This proprety is Readonly.

Type: This property is used to set Type of DateBox. It could be date , datetime and time.

useMaskBehavior: This property is used to Masked user valued to specidifc formate.

Value: This property is used to set or get actual value of DateBox.

These are the important methods of DateBox.

close() – This method is used to close() DropDown Widow.

content() – This method is used to Gets the popup window's content.

getButton(name) – This method is used to get speccifice button based on its name.

open() – This method is used to open dropDown Window.

## **DropDownBox**

acceptCustomValue: This property is used to allow custome value to DropDownBox.

contentTemplate: This method is used to render drop down template.

dataSource: this property is used to Bind data with dropdown.

displayExpr: This property is used for display data filed on DropDownBox;

dropDownButtonTemplate: This method is used for specify template for dropdownbutton.

fieldTemplate: This method is used to provide field Template for dropdownbox.

items[]: This property is used to load Field Items.

onDisposing: This method will call when UI element is Disposing.

onEnterKey: This method is called when UI element is in focused and user press Enter Key.

onCopy: This method is called when UI elment is in focued and user copy Text via right click or pressing ctrl+C.

onCut: This method is called when UI elment is in focued and user cut Text via right click or pressing ctrl+X.

onFocusIn: this method is called when UI Elemet enters in focus state.

onFocusOut: This methos is called when UI element returnev from Focus state.

onInput: This method is called when user input something.

Opened: This property is used to check whether droip down box is opend or not.

openOnFieldClick: This property is used to set whether can open dropdownbox by clicking on textbox or not.

valueExpr: This specify that actual value field from datasource or item for dropdownbox.

valueChangeEvent: this specify that when to call onValueChange event. By feualt it is in when value change. We can set it for "keyup", "blur", "change", "input", and "focusout".

field(): This method returns input elemnt of dropdown box.

## **SelectBox:**

displayValue: This property is used to get display value of dropdownbox.

dropDownButtonTemplate: This field is used to set Template for dropdownbutton.

fieldTemplate: This method is used to define fieldTemplate.

Grouped: This method is used to define whether data is grouped or not.

groupTemplate: this method is used for specifying group template.

items: This property is used to set Items of Select Box.

minSearchLength: This property is used to set minimum lenth of charetec to start searching.

noDataText: This property is used to diplaye message when data source have empty value.

onCustomItemCreating: This propery is called when custom item entered on Text box of select box.

onItemClick: This method called when user click on Item.

onOpened: This method called when DropdownBox opened.

onSelectionChanged: This method called when Item Sekection Change. Difference bewtween OnSelectionChanged and onItemClick is onItemClick is only called when user click on Item via mouese, where onSelectionChanged is also called when user change slection even without opening dropdown.

searchEnabled: This preopery is used to define whether user can serach on selectbox or not.

searchExpr: This propery is used to select data field on which searching will be apply.

searchMode: This proepry is used to set serach mode. It could be 'contains' or 'startswith'.

searchTimeout: This property is user to set Searching Time.

selectedItem: This property is used to get selected item.

showSelectionControls: This property us used to set whethen radiobutoon should be displayed with item for selection or not.

Spellcheck: This property is used to set whther to enable spell check or not.

wrapItemText: This property is used to set whether to wrap Text or not.

Difference between DropDownBox and SelectBox.

|  |  |  |
| --- | --- | --- |
|  | **SelectBox** | **DropDownBox** |
| Accept custom values | [Yes](https://js.devexpress.com/Documentation/Guide/UI_Components/SelectBox/Create_a_User-Defined_Item/) | [Yes](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxDropDownBox/Configuration/#acceptCustomValue) |
| Custom buttons | [Yes](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxSelectBox/Configuration/buttons/) | [Yes](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxDropDownBox/Configuration/buttons/) |
| Keyboard navigation | [Yes](https://js.devexpress.com/Documentation/Guide/UI_Components/SelectBox/Keyboard_Support/) | Requires custom configuration |
| Search box in a drop-down field | - | Requires custom configuration |
| Display groups | [Yes](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxSelectBox/Configuration/#grouped) | Requires custom configuration |
| Search based on user input | [Yes](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxSelectBox/Configuration/#minSearchLength) | Requires custom configuration |
| Multiple selection | - | Requires custom configuration |
| Paging | [Yes](https://js.devexpress.com/Documentation/Guide/UI_Components/SelectBox/Enable_Paging/) | Requires custom configuration |

[Reference Link: https://js.devexpress.com/Documentation/Guide/UI\_Components/SelectBox/How\_to\_Choose\_a\_Drop-Down\_Editor/](https://js.devexpress.com/Documentation/Guide/UI_Components/SelectBox/How_to_Choose_a_Drop-Down_Editor/)

## **NumberBox**

Format: This property is used to formate value.

These chretcters are used to set custome cheretecters.

### **Custom Format String**

A custom format string specifies a format using Unicode Locale Data Markup Language (LDML) patterns. An LDML pattern consists of wildcard characters and characters displayed as is. The following wildcard characters are supported:

**Numeric Formats**

|  |  |
| --- | --- |
| **Format character** | **Description** |
| 0 | A digit. Displays '0' if it is not specified in the UI. |
| # | A digit or nothing. One symbol represents several integer digits, but only one decimal digit. For example, "#0.#" represents "123.4", but not "123.45". |
| . | A decimal separator. Displayed according to the specified locale. |
| , | A group separator. Displayed according to the specified locale. |
| % | The percent sign. Multiplies the input value by 100. If it is enclosed in single quotes ('%'), it only adds this sign to the input value. |
| ; | Separates positive and negative numbers. If there is no explicit negative format, a positive number receives the "-" prefix. |
| Other characters | Any character. Should be placed only at the format string's beginning or end. You can use the special characters above as well (in single quotation marks). |

[For Reference: https://js.devexpress.com/Documentation/ApiReference/Common/Object\_Structures/format/#type](https://js.devexpress.com/Documentation/ApiReference/Common/Object_Structures/format/#type)

<https://js.devexpress.com/Documentation/Guide/Common/Value_Formatting/#Format_UI_Component_Values/Custom_Format_String>

invalidValueMessage: This will display error message when user input invalid inputs.

label: This will use to display label in top of textbox.

labelMode: This will use to set labelMode. It would be number, text and tel.

step: This will use to set step for number.

useLargeSpinButtons: This property is used to set lagre sping Buttons.

## **TextArea:**

inputAttr: it used to provide attributes to input elements.

autoResizeEnabled: It is used to set if TextArea will be resize or not based on contain.

maxLength: It is used to set maxLength of TextArea.

## **TextBox:**

Mask: This is used to allow only specific charectes in textbox.

MaskChar: This is used to specify specific mask character.

MaskInvalidMessage: This is used to specify if mask validation failed.

maskRules: This is used to set Rules for Mask.

A mask can contain the following elements.

|  |  |
| --- | --- |
| **Masking Element** | **Description** |
| 0 | A digit. |
| 9 | A digit or a space. |
| # | A digit, a space, "+" or "-" sign. |
| L | A literal. |
| C | Any character except space. |
| c | Any character. |
| A | An alphanumeric. |
| a | An alphanumeric or a space. |

Mode : This is used to specify Mode of Textbox. It could be 'email', 'password', 'search','tel','text' or 'url'.

showMaskMode: This is used to specify mask Mode. It could be 'always' or 'onFocus'.

Spellcheck: This is used to specify spell chek is visible in UI or not.

useMaskedValue: This is used to specify whether textbox value contains mask character or not.

## **validationGroup:**

It is used to group elements in specific category for validation.

validate(): This method is used to validate all the element of this validation group.

ValidationGroupResult: validate() method return object of ValidationGRoup result.

brokenRules: contains array of broken validation rules.

Complete: contains promise which will fulfil if validate succesffuly for async validation.

isValid: this is specify that is all element is valid or not.

Status: this is used to check if valdation stutus it could be valid, invalid or pending.

Validators: UI components included in the validated group.

## **ValidationSummary:**

Items: Contains array of broken rules messganes.

itemTemplate: specify Item Template to diplay.

onItemClick: Call when user click on specific Item.

validationGroup: Used to check validationGroup for this summary.

## **Validator :**

onValidated: call when validator validate compontans. Contain this object. { brokenRule, brokenRules, isValid, name, status, validationRules, value}

validationGroup: Used to specify specific validation group.

validationRules: This containsobjects of validation Rules for this componants.

These are type of validation rules: RequiredRule | NumericRule | RangeRule | StringLengthRule | CustomRule | CompareRule | PatternRule | EmailRule | AsyncRule.

ValidationRules:

These are the important properties for Validation Rules.

RequiredRule:

message

trim

type : 'required' | 'numeric' | 'range' | 'stringLength' | 'custom' | 'compare' | 'pattern' | 'email' | 'async'

NumericRule:

ignoreEmptyValue:

RangeRule:

ignoreEmptyValue

max

min

reevaluate

StringLengthRule:

same

CustomRule:

validationCallback:

CompareRule:

comparisonTarget

comparisonType '!=' | '!==' | '<' | '<=' | '==' | '===' | '>' | '>='

PatternRule

pattern

EmailRule

# **Actions**

## **Button:**

accessKey: These property is used to access button using keyboard shortcuts. In chrome we have to press key with ‘ALT+’.

icon: This Property is used for defining icon with button text. We can provide icon url, cutome icon class or used DevExtreme icon library.

onClick: This method will call when Button is clicked.

Text: This property is used to set Text For Button.

stylingMode: This property is used to set Styiling mode of Button. We can set 'text' | 'outlined' | 'contained' here.

type: This property is used to define type of Button. We can set 'back' | 'danger' | 'default' | 'normal' | 'success' here.

useSubmitBehavior: This property is used to declare button as Submit button of Form. If onClick is define then it will call before form submitted.

## **FileUploader**

abortUpload: This method to abort Uplodaing file. Function argument will be file and uploadinfo.

Accept: This property describe file typs which will be allow to upload.

[For list of available MIME types: https://www.lifewire.com/file-extensions-and-mime-types-3469109](https://www.lifewire.com/file-extensions-and-mime-types-3469109)

allowCanceling: This property is used to allow user to allow cancle upload while uploading or after upload only if uploadMode is not set to ‘useForm’.

allowedFileExtensions: This property contains array of strings which use to only allow limited file extention.

chunkSize: It is used to specify chunck size when file upload in chunks. File upload mode must be instanlt or useButton.

This is a formate of chunkmetadata.

{

"FileGuid": string,

"FileName": string,

"FileType": string,

"FileSize": long,

"Index": long, // The chunk's index

"TotalCount": long, // The file's total chunk count

}

dialogTrigger: This is used to specify open file upload zone in HTML. We have to pass ID, class name or tag name of Html Element.

dropZone: This is used to specify drag zone in HTML. We have to pass ID, class name or tag name of Html Element.

invalidFileExtensionMessage: This is used to specify InvalidFileExtensionMessage.

invalidMaxFileSizeMessage: This is used to invalid max file size message.

invalidMinFileSizeMessage: : This is used to invalid min file size message.

labelText: This is use to specify label text for default drop area.

maxFileSize: This is used to specify MaxFileSize only if uploadmode is "instantly" or "useButtons".

minFileSize: This is used to specify minFileSize only if uploadmode is “instantly” or “useButtons”. Multiple: This is used to specify user can select multiple files or not.

onBeforeSend: This will call before uploading begin.

onDropZoneEnter: Will call when file enter in DropZone.

onDropZoneLeave: Will call when file leave in DropZone.

onFilesUploaded: Will Call when FileUploaded.

onProgress: Will call whle file uploading.

onUploadAborted: Will call after Upload Aborted.

onUploaded: Will call after File Uploaded.

onUploadError: Will call if Upload have Error.

onUploadStarted: Will call when upload started.

readyToUploadMessage: Will display Message when File is ready to uload. This property makes sense only if the uploadMode property is set to "useButtons".

selectButtonText: use to specify text of upload Button.

showFileList: Will be use to Display list of Files.

uploadAbortedMessage: Use to specify File Upload Fail message. This property is only available if the uploadMode property is set to "instantly".

uploadButtonText: The text displayed on the button that starts uploading. The property makes sense only if the uploadMode property is set to "useButtons" or "instantly".

uploadChunk: This method is used to upooad chunk.

uploadCustomData: This will use to transfer custome data.

uploadedMessage: “This will use to set Upoladed Messag.”

uploadFailedMessage: Used to set upload Failed Message.

uploadFile: Method Will upload file.

uploadHeaders: Will contain upload Header.

uploadMethod

uploadMode: Specifies how the UI component uploads files. Accepted Values: 'instantly' | 'useButtons' | 'useForm'.

uploadUrl: Specifies a target Url for the upload request. The property makes sense only if the uploadMode property is set to "useButtons" or "instantly".

Methods:

abortUpload():abortUpload(file) abortUpload(fileIndex)

The **abortUpload** method works differently in the following [upload modes](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxFileUploader/Configuration/#uploadMode):

* **useForm**: The method is not supported in this mode.
* **useButtons**: Cancels the file upload and makes the file available for upload.
* **instantly**: Cancels the file upload.

# **Data Layers:**

## **ArrayStore:**

Array Store is used to load Data in Memory in Array Form.

The ArrayStore is immutable. We cannot change its configuration at runtime. However, you can use its methods to manipulate it.

Properties and Methods:

Data: It is used to store Data in ArrayStore.

errorHandler: It will executed when ArrayStore Throws Error.

Key: It is used to specify Key Element.

onInserted: It will call after new value is inserted.

onInserting: It will call after new value is inserting.

onLoaded: It will call after new data is loaded.

onModified: It will call after array store items is modified.

onModifying: It will call after array store iterms is modifing

onPush: It will call after new value is inserted via push method.

onRemoved: It will call after existing value is removed.

onRemoving: It will call after items is removing.

onUpdated: It will call after existing item is updated.

onUpdating: it wil call after existing items is updating.

byKey(key): It will return object by specific Key.

clear(): It will clear Data from Array Store.

createQuery(): It will create new Query object for

insert(values): It will insert new value into ArrayStore.

key(): It will return Key field of Array.

keyOf(obj): It will return Key of particular object.

load(), load(options): It will start loading data.

push(changes): It wil

remove(key): It will remove specific Item from Arrayt based on Key.

totalCount(options): It will return total length of an array.

update(key, values): It will update value of particular item based on array.

## **Query:**

It use to perform chainbale Query on Data.

We can Create Query By Calling query method of DevExpress.data.query method. Which accept array as one argument or url and queryOption as two parametes for creating Query instance that accesses a remote data service using its URL.

Example:

DevExpress.data.query([10, 20, 50, 40, 30]);

DevExpress.data.query("http://mydomain.com/MyDataService", queryOptions);

Methods:

aggregate(seed, step, finalize)

It is User to calculate or perform some operatrion for all data.

In Paramater seed represent initial value. step is a method which will call for each items. Where is first paramters contains value of last return value of method and second parameter contain current data item of an Array.

Finalize is also method which will call after step method is call for each items.

This function will return promise object.

aggregate(step)

It will call step method for all items.

Avg(),avg(getter)

This method will return promiose which contains average value of an Array. It must be apply with numraci array or we can use getter for specifying numeric column.

count(): This method is used to count all the object of an Array.

filter(criteria): This method is used to filter Data. Where criteria contains 3 arguments where first argument on which we want to applt comparition, second argument contains comparition opearator. The available operators are: "=", "<>", ">", ">=", "<", "<=", "startswith", "endswith", "contains", "notcontains". and third argument contains value or objct with comparition will apply.

We can also pass an Array which have similar these 3 argument.

By using Array we can apply multiple compration.

filter(predicate): This method is also used to filter Data. Where predicate accept an function which will used to customize filete. A function must return an boolen value for filter value.

Both above function will return.

groupBy: It us used to group data based on fiald. It return Query Object.

max(), max(getter): It is used to find max value from Data. Its return promise.

min(), min(getter): It is used to find min value from Data. Its return promise.

select(getter), Select(array[getter]): It is used to particular fialds from Data. Its Query object promise.

slice(skip, take): I is used to slice Array value. First parameter contains starting of index. And second parameter contains number of values to take.

sortBy(getter), sortBy(getter, desc): It is used to sort data based on column value.

sum(),sum(getter) This value is used to sum() value of particular column.

thenBy(‘’): This is used to sort for multiple fialds.

## **Data Source:**

It is an object that provide an API for accessing data from store.

It must be Dispose if it is used outside of UI element.

customQueryParams: It is used to passed custome paramters.

Expand: It is used to expand field in navigation.

Filter: It is used to filter data.

Group: It is used to specify group filad. Via string, object(selector, desc:true), array<objects>, or function.

Map: It is used to map an data items for eaxh elements.

onChanged: It will call after data is loaded. Function para,ter changes: Appears only when the push(changes) method is called and the reshapeOnPush property is false.

onLoadError(error): When load fails.

pageSize: Specifies the maximum number of data items per page. Applies only if paginate is true.

Paginate: Specifies whether the DataSource loads data items by pages or all at once. Defaults to false if group is set; otherwise, true.

postProcess: Specifies a post processing function. When the paginate property is enabled, the postProcess function handles only data available for the selected page. If you need to access all data, process data before it is passed to the store.

reshapeOnPush: Specifies whether to reapply sorting, filtering, grouping, and other data processing operations after receiving a push.

searchExpr: In most cases, you should pass the name of a field by whose value data items are searched. Assign an array of field names to this property if you need to search elements by several field values.

searchValue: Specifies the value to which the search expression is compared.

Methods:

filter(): get filterExpr. filter(filterExpr) set filterExpr

group() : Gets the group property's value. ,group()

These are also useful method for get and set properties of Datasource.

isLastPage(),isLoaded(),isLoading(),items(),key(),load(),loadOptions(),pageIndex(),pageIndex(newIndex), pageSize(),pageSize(value), paginate(),paginate(value), reload(),requireTotalCount(),requireTotalCount(value), searchExpr(),searchExpr(expr), searchOperation(),searchOperation(op), searchValue(value), select(),select(expr), sort(),sort(sortExpr)¸ store(), totalCount(),

totalCount()

Gets the number of data items in the store after the last load() operation without paging. Takes effect only if requireTotalCount is true.

## **localStore:**

Only difference between arraystore and localstore is data is stored in localstore of browser.

flushInterval: This property set interval for changes. Only if immediate is false.

## **CustomeStore:**

The CustomStore enables you to implement custom data access logic for consuming data from any source.

The CustomStore's implementation depends on whether data is processed on the client or server. For client-side data processing, implement the load function to load data from the data source.

### **Load Data in Raw Mode**

Loading data in raw mode allows you to configure the CustomStore more easily. You can use it only if all data shaping operations are supposed to be performed on the client. In raw mode, the [load](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Configuration/#load) function should get raw, unprocessed data from the server, and the CustomStore will perform data shaping automatically, without any input from you. To switch to the raw mode, assign "raw" to the [loadMode](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Configuration/#loadMode) property.

* var store = new DevExpress.data.CustomStore({
* loadMode: "raw",
* load: function() {
* return $.getJSON("url/to/the/resource");
* }
* });

Note that you are not required to implement the [byKey](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Configuration/#byKey) and [totalCount](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Configuration/#totalCount) functions in raw mode, since they will be evaluated based on the results of the **load** function. If, however, you do implement them, your implementation will take precedence over the default one.

Once loaded, data is stored in the cache. If you need to clear the cache at some point, call the [clearRawDataCache()](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Methods/#clearRawDataCache) method.

* store.clearRawDataCache();

To switch data caching off, assign **false** to the [cacheRawData](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Configuration/#cacheRawData) property. Note that in this case, the CustomStore will reload all data on every call of the **load**, **byKey** and **totalCount** functions.

* var store = new DevExpress.data.CustomStore({
* // ...
* cacheRawData: false
* });

Since the CustomStore loads all data in raw mode at once, we do not recommend using it with large amounts of data. If you notice a decrease in the CustomStore performance in raw mode, consider delegating some or all data shaping operations to the server and implementing the remaining operations in the [load](https://js.devexpress.com/Documentation/ApiReference/Data_Layer/CustomStore/Configuration/#load) function yourself.

[Reference Link: https://js.devexpress.com/Documentation/Guide/Data\_Binding/Data\_Source\_Examples/#Custom\_Sources/Load\_Data\_in\_Raw\_Mode/](https://js.devexpress.com/Documentation/Guide/Data_Binding/Data_Source_Examples/#Custom_Sources/Load_Data_in_Raw_Mode/)

Propertiews and methods:

byKey: function(key): custome implementation of byKey for getting specific key value from api or data.

cacheRawData: will specify that whether raw data should be cached or get all data from server everytime. Onlit if loadMode is “row”.

Insert: function (values): for custome implmantation of Insert.

Load: function (options): custome implementation of load.

loadMode: specify loadmod from data or api. Value could be 'processed' | 'raw'.

Specify this property depending on the behavior you implemented for the load function. If this function sends data shaping properties to the server and fetches processed data, then loadMode should be "processed". If the load function simply fetches raw, unprocessed data from the server, set loadMode to "raw". In this case, the raw data will be processed on the client automatically.

remove : function(key) for custome implementation of Remove.

totalCount : function([filter,group]): custome implementation of totalcount:

update : function(filter,group): custome implementation of update:

useDefaultSearch: Specifies whether the store combines the search and filter expressions. Defaults to true if the loadMode is "raw" and false if it is "processed".

# **Form**

alignItemLabels: It is used to align labels in form.

alignItemLabelsInAllGroups: It is used to align labels in form.

colCount: number of columns to display in layout.

colCountByScreen: number of by screen.

formData: Provides the Form's data. Gets updated every time form fields change.

## **Items:**

Holds an array of form items.

Type: Array<Simple Form Item | Group Form Item | Tabbed Form Item | Empty Form Item | Button Form Item>

### **SimpleItem**

It is a simple item for Form.

#### **Properties:**

colSpan: “Specifies the number of columns spanned by the item.”

cssClass: “Specifies a CSS class to be applied to the form item.”

dataField: “Specifies the path to the formData object field bound to the current form item.”

editorOptions: “Configures the form item's editor.”

editorType: “'dxAutocomplete' | 'dxCalendar' | 'dxCheckBox' | 'dxColorBox' | 'dxDateBox' | 'dxDropDownBox' | 'dxHtmlEditor' | 'dxLookup' | 'dxNumberBox' | 'dxRadioGroup' | 'dxRangeSlider' | 'dxSelectBox' | 'dxSlider' | 'dxSwitch' | 'dxTagBox' | 'dxTextArea' | 'dxTextBox'”

helpText: Specifies the help text displayed for the current form item.

isRequired: “Specifies whether the current form item is required.”

itemType: “Specifies the item's type. Set it to "simple" to create a simple item.”

Accepted Values: 'empty' | 'group' | 'simple' | 'tabbed' | 'button'

Label: Specifies properties for the form item label.

Name: Specifies a name that identifies the form item.

Template: A template that can be used to replace the default editor with custom content.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **component** | [Form](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Widgets/dxForm/) | The Form instance. |
| **dataField** | [String](http://www.w3schools.com/js/js_strings.asp) | The item's [dataField](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxForm/Item_Types/SimpleItem/#dataField). |
| **editorOptions** | [Object](http://www.w3schools.com/js/js_objects.asp) | The item editor's [configuration](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxForm/Item_Types/SimpleItem/#editorOptions). |
| **editorType** | [String](http://www.w3schools.com/js/js_strings.asp) | The editor's [type](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxForm/Item_Types/SimpleItem/#editorType). |
| **name** | [String](http://www.w3schools.com/js/js_strings.asp) | The item's [name](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxForm/Item_Types/SimpleItem/#name). |

visibleIndex : Specifies the sequence number of the item in a form, group or tab.

### **GroupItem:**

A group form item is a section consisting of a caption and child form items. You can customize the layout properties for each group separately.

alignItemLabels: Specifies whether or not all group item labels are aligned.

Caption: Specifies the group caption.

colCount: The count of columns in the group layout.

colCountByScreen: Specifies the relation between the screen size qualifier and the number of columns in the grouped layout.

colSpan: Specifies the number of columns spanned by the item.

Items:

### **TabbedItem:**

An item representing a tabbed container for other form items.

### tabPanelOptions

Holds a configuration object for the TabPanel UI component used to display the current form item.

tabs[]: An array of tab configuration objects. Type: Array<Object>

### **EmptyItem:**

An empty item used to add a space between neighboring items.

### **ButtonItem:**

A button form item, as the name implies, consists of a button that does a custom action. You can customize the button's appearance and position on the form.

buttonOptions

Configures the button.

horizontalAlignment: 'center' | 'left' | 'right'

verticalAlignment: 'bottom' | 'center' | 'top'

## Form Remaining properties.

labelLocation: 'left' | 'right' | 'top'.

minColWidth: The minimum column width used for calculating column count in the form layout. Applies only if colCount property is "auto". Default Value: 200

onEditorEnterKey: A function that is executed when the Enter key has been pressed while an editor is focused.

onFieldDataChanged: A function that is executed when the value of a formData object field is changed.

optionalMark: The text displayed for optional fields. Default Value: 'optional'.

readOnly: Specifies whether all editors on the form are read-only. Applies only to non-templated items.

requiredMark: The text displayed for required fields.

requiredMessage: '{0} is required'

screenByWidth: Specifies a function that categorizes screens by their width.

The UI component uses the following size qualifiers to categorize screens by width:

|  |  |
| --- | --- |
| **Size Qualifier** | **Description** |
| xs | Stands for "extra small". Screens with width less than 768 pixels. |
| sm | Stands for "small". Screens with width between 768 and 992 pixels. |
| md | Stands for "medium". Screens with width between 992 and 1200 pixels. |
| lg | Stands for "large". Screens with width more than 1200 pixels. |

Implement the **screenByWidth** function to change the relation between a size qualifier and screen width. This function accepts the screen width and should return a size qualifier. The following code shows the function's default implementation that you can customize.

**scrollingEnabled:** A Boolean value specifying whether to enable or disable form scrolling.

**showColonAfterLabel:** Specifies whether or not a colon is displayed at the end of form labels.

**showOptionalMark:** Specifies whether or not the optional mark is displayed for optional fields.

**showRequiredMark:** Specifies whether or not the required mark is displayed for required fields.

**showValidationSummary:** Specifies whether or not the total validation summary is displayed on the form.

**validationGroup:** Gives a name to the internal validation group.

**getButton(name):** Gets a button's instance.

**getEditor(dataField):** Gets an editor instance. Takes effect only if the form item is visible.

**getInstance(element):** Gets the instance of a UI component found using its DOM node.

**itemOption(id):** Gets a form item's configuration.

**itemOption(id, option, value):** Updates the value of a single item property.

**itemOption(id, options):** Updates the values of several item properties.

**resetOption(optionName):** Resets a property to its default value.

**resetValues():**Resets the editor's value to undefined.

**updateData(data):** Merges the passed data object with formData. Matching properties in formData are overwritten and new properties added.

**updateData(dataField, value):** Updates a formData field and the corresponding editor.

**updateDimensions():** Updates the dimensions of the UI component contents.

## **RadioGroup:**

dataSource: Binds the UI component to data.

displayExpr: Specifies the data field whose values should be displayed.

items[]: array of objects which Contains Radiobuttons name and values.

layout: layout either horzantal and verticle.

valueExpr: define value exprestion from items.

# **DataGrid:**

The DataGrid is a UI component that represents data from a local or remote source in the form of a grid. This UI component offers such basic features as sorting, grouping, filtering, as well as more advanced capabilities, like state storing, client-side exporting, master-detail interface, and many others.

**allowColumnReordering:**

Specifies whether a user can reorder columns.

Initially, columns appear in the order specified by the columns array. If you skip specifying this array, columns will mirror the order of fields in the first object from the dataSource. You can allow a user to reorder columns at runtime by setting the allowColumnReordering property to true.

**allowColumnResizing**

Specifies whether a user can resize columns.

By default, the width of each column depends on the width of the UI component and the total number of columns. You can allow a user to resize the columns at runtime by setting the allowColumnResizing property to true.

**autoNavigateToFocusedRow**

Automatically scrolls to the focused row when the focusedRowKey is changed. Incompatible with infinite scrolling mode.

**cacheEnabled**

Specifies whether data should be cached.

When this property is set to true, data loaded once is saved in cache. Then, the UI component takes data from this cache when performing such operations as sorting, grouping, paging, etc. Caching is helpful when the data source takes significant time to load. But, consider disabling it for frequently changing data sources.

To update data in cache, call the refresh() method of the UI component or the load() method of the DataSource.

**cellHintEnabled**

Enables a hint that appears when a user hovers the mouse pointer over a cell with truncated content.

**columnAutoWidth**

Specifies whether columns should adjust their widths to the content.

**columnChooser**

Configures the column chooser.

The column chooser allows a user to hide columns at runtime. To enable it, assign true to the columnChooser.enabled property.

**allowSearch:**

Specifies whether searching is enabled in the column chooser.

**emptyPanelText**

Specifies text displayed by the column chooser when it is empty.

**enabled**

Specifies whether a user can open the column chooser.

**height**

Specifies the height of the column chooser.

**mode**

Specifies how a user manages columns using the column chooser.

Accepted Values: 'dragAndDrop' | 'select'

In drag and drop mode, a user moves column headers to and from the column chooser by drag and drop. In select mode, the user selects column headers in the column chooser using check boxes. The select mode is designed primarily for touch-enabled devices. On mouse-equipped platforms, either mode is suitable.

**searchTimeout**

Specifies a delay in milliseconds between when a user finishes typing in the column chooser's search panel, and when the search is executed.

**title**

Specifies the title of the column chooser.

**width**

Specifies the width of the column chooser.

**columnFixing:**

When the width of all columns exceeds the UI component width, horizontal scrolling appears. If specific columns should be on screen constantly regardless of how far the UI component is scrolled, allow a user to fix them at runtime using the context menu. For this, set the columnFixing.enabled property to true.

Enabled: Enables column fixing. - false

texts

Contains properties that specify texts for column fixing commands in the context menu of a column header.

**Fix**: Specifies text for the context menu item that fixes a column. – Fix

**leftPosition: Specifies text for the context menu subitem that fixes a column to the left edge of the UI component. - 'To the left'**

rightPosition - Specifies text for the context menu subitem that fixes a column to the right edge of the UI component. - 'To the right'

**unfix -** Specifies text for the context menu item that unfixes a column. ‘Unfix’

**columnHidingEnabled**

Specifies whether the UI component should hide columns to adapt to the screen or container size. Ignored if allowColumnResizing is true and columnResizingMode is "widget". – false

**columnMinWidth**

Specifies the minimum width of columns.

**columnResizingMode**

Specifies how the UI component resizes columns. Applies only if allowColumnResizing is true. 'nextColumn' | 'widget'

**nextColumn**

When a user resizes a column, the width of the next column changes.

**widget**

When a user resizes a column, the width of the UI component changes.

This mode is ignored if you specify the width of any column in percent.

repaintChangesOnly:

## **Columns[]:**

By default, a column is created for each field of a data source object, but in most cases, it is redundant. To specify a set of columns to be created in a grid, assign an array specifying these columns to the columns property. Each grid column is represented in this array by an object containing column settings or by a data source field that this column is bound to. Detailed information on specifying grid columns is given in the Columns Overview article.

Column properties define the behavior and appearance of a grid column. One of the other capabilities allows you to control the sorting of column values using the allowSorting and sortOrder properties, apply a filter to grid records using the allowFiltering and filterOperations properties, and group grid records using the allowGrouping and groupIndex properties. In addition, you can change the visibility and width of a column using corresponding properties.

To get or set a property or several properties for a column at runtime, use the columnOption method with the required arguments.

**alignment**

**Aligns the content of the column. – undefined**

|  |  |
| --- | --- |
| **dataType** | **alignment** |
| ***'number'*** | ***'right'*** |
| ***'boolean'*** | ***'center'*** |
| ***'string'*** | ***'left'*** |
| ***'date'*** | ***'left'*** |
| ***'datetime'*** | ***'left'*** |

**allowEditing**

**undefined | 'center' | 'left' | 'right'**

**allowExporting**

**Specifies whether data from this column should be exported. Applies only if the column is visible.**

**allowFiltering**

**Specifies whether data can be filtered by this column. Applies only if filterRow.visible is true.**

**allowFixing**

**Specifies whether a user can fix the column at runtime. Applies only if columnFixing.enabled is true.**

**allowGrouping**

**Specifies whether the user can group data by values of this column. Applies only when grouping is enabled.**

**allowHeaderFiltering**

Specifies whether the header filter can be used to filter data by this column. Applies only if headerFilter.visible is true. By default, inherits the value of the allowFiltering property.

**allowHiding**

Specifies whether a user can hide the column using the column chooser at runtime. Applies only if columnChooser.enabled is true.

allowReordering

Specifies whether this column can be used in column reordering at runtime. Applies only if allowColumnReordering is true.

allowResizing

Specifies whether a user can resize the column at runtime. Applies only if allowColumnResizing is true.

allowSearch

Specifies whether this column can be searched. Applies only if searchPanel.visible is true. Inherits the value of the allowFiltering property by default.

allowSorting

Specifies whether a user can sort rows by this column at runtime. Applies only if sorting.mode differs from "none"

autoExpandGroup

Specifies whether groups appear expanded or not when records are grouped by a specific column. Setting this property makes sense only when grouping is allowed for this column.

buttons[]

Allows you to customize buttons in the edit column or create a custom command column. Applies only if the column's type is "buttons".

Type: Array<String | Column Button>

Accepted Values: 'cancel' | 'delete' | 'edit' | 'save' | 'undelete'

calculateCellValue

Calculates custom cell values. Use this function to create an unbound data column.

|  |  |
| --- | --- |
| **Feature** | **Action that enables it** |
| Editing | Implement the [setCellValue](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#setCellValue) function. |
| Sorting | Set the [allowSorting](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#allowSorting) property to **true**. |
| Filtering | Set the [allowFiltering](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#allowFiltering) property to **true**. |
| Searching | Set the [allowSearch](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#allowSearch) property to **true**. |
| Grouping (**DataGrid** only) | Set the [allowGrouping](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#allowGrouping) property to **true**. |

To invoke the default behavior, call the **defaultCalculateCellValue** function and retu

**calculateFilterExpression**

Specifies the column's custom rules to filter data.

Type: Function

Function parameters:

filterValue: any

A user input value.

Contains an array if the selectedFilterOperation is one of the following: "between", "anyof", "noneof".

selectedFilterOperation: String

A selected filter operation.

target: String

A UI element used to filter data.

Possible values: "filterRow", "headerFilter", "filterBuilder", "search".

Return Value: Filter Expression

A filter expression.

If you filter data remotely, the filter expression must not contain functions.

The calculateFilterExpression function should return a filter expression. A basic filter expression has the following format:

**calculateGroupValue**

Sets custom column values used to group grid records

calculateSortValue

Calculates custom values used to sort this column.

Type: String | Function

Function parameters:

rowData: Object

The data of the row to which the cell belongs.

**caption**

Specifies a caption for the column.

**cellTemplate**

Specifies a custom template for data cells.

Type: template

Template Data:

Name Type Description

column DataGrid Column The column's properties.

columnIndex Number The index of the cell's column.

For more information on how this index is calculated, refer to the Column and Row Indexes topic.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **column** | [DataGrid Column](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Widgets/dxDataGrid/Configuration/columns/) | The column's properties. |
| **columnIndex** | [Number](http://www.w3schools.com/js/js_numbers.asp) | The index of the cell's column. For more information on how this index is calculated, refer to the [Column and Row Indexes](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Columns/Column_and_Row_Indexes/) topic. |
| **component** | [DataGrid](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Widgets/dxDataGrid/) | The UI component's instance. |
| **data** | [Object](http://www.w3schools.com/js/js_objects.asp) | The data of the row to which the cell belongs. |
| **displayValue** | any | The cell's display value. Differs from the **value** field only when the column uses [lookup](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/lookup/) or [calculateDisplayValue](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#calculateDisplayValue). |
| **oldValue** | any | The cell's previous raw value. Defined only if [repaintChangesOnly](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/#repaintChangesOnly) is **true**. |
| **row** | [DataGrid Row](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Widgets/dxDataGrid/Row/) | The cell's row. |
| **rowIndex** | [Number](http://www.w3schools.com/js/js_numbers.asp) | The index of the cell's row. Begins with 0 on each page. Group rows are included. For details on row indexes, see the [Column and Row Indexes](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Columns/Column_and_Row_Indexes/) topic. |
| **rowType** | [String](http://www.w3schools.com/js/js_strings.asp) | The row's [type](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Row/#rowType). |
| **text** | [String](http://www.w3schools.com/js/js_strings.asp) | **displayValue** after applying [format](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#format) and [customizeText](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#customizeText). |
| **value** | any | The cell's raw value. |
| **watch** | [Function](https://www.w3schools.com/js/js_functions.asp) | Allows tracking a variable and performing actions when it changes. Applies when [repaintChangesOnly](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/#repaintChangesOnly) is **true**. This function has the following parameters:   * **getter(data)**: Function A function that returns the variable that should be tracked. * **handler(newValue)**: Function |

**customizeText**: Customizes the text displayed in column cells.

**bject structure:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| **groupInterval** | [String](http://www.w3schools.com/js/js_strings.asp)   |  [Number](http://www.w3schools.com/js/js_numbers.asp) | Indicates how header filter values were combined into groups. Available if **target** is *"headerFilter"*. See the **headerFilter**.[groupInterval](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/headerFilter/#groupInterval) property's description for possible values. |
| **target** | [String](http://www.w3schools.com/js/js_strings.asp) | The UI element where the **customizeText** function was called: *"row"*, *"filterRow"*, *"headerFilter"*, *"search"*, *"filterPanel"*, or *"filterBuilder"*. |
| **value** | [String](http://www.w3schools.com/js/js_strings.asp)   |  [Number](http://www.w3schools.com/js/js_numbers.asp)   |  [Date](http://www.w3schools.com/js/js_dates.asp) | The cell's raw value. |
| **valueText** | [String](http://www.w3schools.com/js/js_strings.asp) | The [fomatted](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#format) value converted to a string. |

**dataField**

Binds the column to a field of the dataSource.

The columns array can contain column objects and data field names as strings. If you use column objects, specify the dataField property to bind the object to a column from a data source:

**NOTE**

Review the following notes about data binding:

* If you create an unbound column (use the [calculateCellValue](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#calculateCellValue) function), specify the **columns[]**.[name](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#name) property instead of **dataField**.
* Data field names cannot be equal to this and should not contain the following characters: ., :, [, and ].
* Column caption is generated from the **dataField** value. If you want to use a custom caption, specify it in the [caption](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#caption) property. Unlike **dataField**, **caption** can contain any characters.

datatype

editCellTemplate

Specifies a custom template for data cells in editing state.

Casts column values to a specific data type.

**Type:**

[String](http://www.w3schools.com/js/js_strings.asp)

**Default Value:** undefined

**Accepted Values:** 'string' | 'number' | 'date' | 'boolean' | 'object' | 'datetime'

If a data field provides values of one data type, but the UI component should cast them to another, specify the proper type in this property. In the following code, values of the ID and hireDate fields are cast to numeric and date data types, respectively.

**showBorders**

Specifies whether the outer borders of the UI component are visible.

**showBorders**: false,

**showColumnLines**: false,

**showRowLines**: false,

**rowAlternationEnabled**: true ,

**editorOptions**

Configures the default UI component used for editing and filtering in the filter row.

In this object, you can specify the default UI component's properties (except onValueChanged, which you can specify in onEditorPreparing).

The default editor UI component depends on the column configuration. The following table illustrates the dependency.

**encodeHtml**

Specifies whether HTML tags are displayed as plain text or applied to the values of the column.

**falseText**

In a boolean column, replaces all false items with a specified text. Applies only if showEditorAlways property is false.

**encodeHtml**

Specifies whether HTML tags are displayed as plain text or applied to the values of the column.

**falseText**

In a boolean column, replaces all false items with a specified text. Applies only if showEditorAlways property is false.

**filterOperations**

Specifies available filter operations. Applies if allowFiltering is true and the filterRow and/or filterPanel are visible.

Accepted Values: '=' | '<>' | '<' | '<=' | '>' | '>=' | 'contains' | 'endswith' | 'isblank' | 'isnotblank' | 'notcontains' | 'startswith' | 'between' | 'anyof' | 'noneof'

filterType ('exclude' | 'include')

Specifies whether a user changes the current filter by including (selecting) or excluding (clearing the selection of) values. Applies only if headerFilter.visible and allowHeaderFiltering are true.

lookup**:**

**allowClearing**

Specifies whether to display the Clear button in lookup column cells while they are being edited.

**dataSource**

Specifies the data source for the lookup column.

**displayExpr**

Specifies the data source field whose values must be displayed.

**valueExpr**

Specifies the data field whose values should be replaced with values from the displayExpr field.

**groupIndex**

Specifies the index of a column when grid records are grouped by the values of this column.

headerCellTemplate(): uysed to define header template

### formItem

**editorOptions**

**editorType**

### HeaderFilter

**allowSearch**: true,

**groupInterval**: 'day' | 'hour' | 'minute' | 'month' | 'quarter' | 'second' | 'year' | number

**searchMode**

## Scrolling

Scrolling allows browsing data outside the UI component's viewport. The following scrolling modes are available in the DataGrid:Scrolling allows browsing data outside the UI component's viewport. The following scrolling modes are available in the DataGrid:

**columnRenderingMode**

Specifies the rendering mode for columns. Applies when columns are left outside the viewport. Requires the columnWidth, columnAutoWidth, or width (for all columns) property specified.

### mode

Specifies the scrolling mode.

**Standard**

A user scrolls a single page only. Scrolling is available only if all the page's rows do not fit into the UI component's height. In this mode, the pager performs the main navigation and scrolling is auxiliary. If paging is disabled, the UI component loads all rows simultaneously which can reduce the UI component's performance. In this case, we recommend that you use another scrolling mode.

**Virtual**

Rows are loaded when they get into the viewport and removed once they leave it. If the rows take time to be loaded and rendered, they display gray boxes. Rendering optimization can reduce rendering time and remove gray boxes. In this mode, users can move to any page instantly.

**Infinite**

Each next page is loaded once the scrollbar reaches the end of its scale. In this mode, users scroll data gradually from the first to the last page.

Set the grouping.allowCollapsing property to false when using infinite scrolling in conjunction with grouping.

Regardless of the scrolling mode, you can use the paging.pageSize property to specify the number of rows on a page.

**preloadEnabled**

Specifies whether the UI component should load adjacent pages. Applies only if scrolling.mode is "virtual" or "infinite".

**rowRenderingMode**

Specifies the rendering mode for loaded rows.

"standard"

Renders all the loaded rows at once.

"virtual"

Renders only those rows that get into the viewport. Use this mode if the pageSize is large. Rows that are being currently rendered can be shown as grey boxes.

**scrollByContent**

Specifies whether a user can scroll the content with a swipe gesture. Applies only if useNative is false.

## Paging

Paging allows the UI component to load data in portions instead of loading it simultaneously. To enable paging, set the paging.enabled property to true.

Users can switch between pages and change paging settings using the pager or they can scroll the pages. Paging settings apply with any scrolling mode.

**enabled**

Enables paging

**pageIndex**

Specifies the page to be displayed using a zero-based index.

**pageSize**

**Specifies the page size.**

pager**:**

**Configures the pager.**

The pager is an element that allows users to navigate through pages and change their size at runtime. The pager consists of the page navigator and several optional elements: the page size selector, navigation buttons, and page information.

**allowedPageSizes**

Specifies the available page sizes in the page size selector.

**infoText**

Specifies the page information text.

**showInfo**

Specifies whether to show the page information.

**showNavigationButtons**

Specifies whether to show navigation buttons.

**showPageSizeSelector**

Specifies whether to show the page size selector.

**visible**

Specifies whether the pager is visible.

## toolbar

Configures the toolbar.

**disabled**

Specifies whether the toolbar responds to user interaction.

items**[]**

Configures toolbar items.

Accepted Values: 'addRowButton' | 'applyFilterButton' | 'columnChooserButton' | 'exportButton' | 'groupPanel' | 'revertButton' | 'saveButton' | 'searchPanel'

**html**

Specifies the HTML markup to be inserted into the item element.

**locateInMenu**

Specifies when to display an item in the toolbar's overflow menu.

Accepted Values: 'always' | 'auto' | 'never'

**location:**

Specifies a location for the item on the toolbar

Accepted Values: 'after' | 'before' | 'center

**menuItemTemplate**

Specifies a template that should be used to render a menu item.

**name**:

'addRowButton' | 'applyFilterButton' | 'columnChooserButton' | 'exportButton' | 'groupPanel' | 'revertButton' | 'saveButton' | 'searchPanel'

**option:**

**showText:**

toolbar.visibleDescription**true**The toolbar is always visible.**false**The toolbar is always hidden.**undefined**The toolbar is visible only if there are [items](https://js.devexpress.com/Documentation/ApiReference/UI_Components/dxDataGrid/Configuration/toolbar/items/) to display.

## editing**:**

The UI component can allow a user to add, update and delete data. To control which of these operations are allowed, use the allowAdding, allowUpdating and allowDeleting properties. Editing can be carried out in different modes, which are detailed in the mode property's description.

**allowAdding**

Specifies whether a user can add new rows.

**allowDeleting**

Specifies whether a user can delete rows. It is called for each data row when defined as a function.

**allowUpdating**

Specifies whether a user can update rows. It is called for each data row when defined as a function.

**changes**

Pending row changes.

The changes array can be changed from the UI, with native JavaScript methods, or with UI component methods (addRow, editRow, editCell, deleteRow). However, objects with the "insert" type cannot be added with native JavaScript methods. We plan to add this functionality in future releases.

**confirmDelete**

Specifies if confirmation is required when a user deletes a row.

**editColumnName**

The name of a column being edited. Applies only if editing.mode is "cell" or "batch".

**editRowKey**

The key(s) of a row being edited.

**form**

Configures the form. Used only if editing.mode is "form" or "popup".

**popup**

Configures the popup. Used only if editing.mode is "popup".

Type: Form Configuration

Default form editors depend on the columns' configuration. If the generated form does not meet your requirements, and you need to reorganize form items or set other form properties, specify it in the form object. To link a form item with a grid column, assign identical values to the form.items.dataField and columns.dataField properties.

### mode**:**

Specifies how a user edits data.

'batch' | 'cell' | 'row' | 'form' | 'popup'

* [**Row**](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Editing/#User_Interaction/Row_Mode)  
  A user edits one row at a time. The UI component saves changes when the row leaves the editing state.

[View Demo](https://js.devexpress.com/Demos/WidgetsGallery/Demo/DataGrid/RowEditingAndEditingEvents)

* [**Batch**](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Editing/#User_Interaction/Batch_Mode)  
  A user edits data cell by cell. The UI component does not save changes until a user clicks the global *"Save"* button.
* selectTextOnEditStart: true,false
* startEditAction click-dblclick

[View Demo](https://js.devexpress.com/Demos/WidgetsGallery/Demo/DataGrid/BatchEditing/)

* [**Cell**](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Editing/#User_Interaction/Cell_Mode)  
  Differs from the batch mode in that the UI component saves changes when the cell leaves the editing state.
* startEditAction click-dblclick
* selectTextOnEditStart: true,false

[View Demo](https://js.devexpress.com/Demos/WidgetsGallery/Demo/DataGrid/CellEditingAndEditingAPI/)

* [**Form**](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Editing/#User_Interaction/Form_Mode)  
  On entering the editing state, a row becomes a form with editable fields. The UI component saves changes after a user clicks the *"Save"* button.

[View Demo](https://js.devexpress.com/Demos/WidgetsGallery/Demo/DataGrid/FormEditing/)

* [**Popup**](https://js.devexpress.com/Documentation/20_2/Guide/UI_Components/DataGrid/Editing/#User_Interaction/Popup_Mode)  
  Differs from the form mode in that the form with editable fields is placed in a popup window.

[View Demo](https://js.devexpress.com/Demos/WidgetsGallery/Demo/DataGrid/PopupEditing/)

## grouping

**allowCollapsing**

Specifies whether the user can collapse grouped records in a grid or not.

**autoExpandAll**

Specifies whether groups appear expanded or not. – true

**contextMenuEnabled**

Set this property to true to enable grouping using the context menu. In this case, to group/ungroup data, the user right-clicks a column header and chooses the required item from the appeared context menu.

**expandMode**

Specifies the event on which a group will be expanded/collapsed.

Default Value: 'buttonClick', 'rowClick' (mobile devices)

### texts

groupByThisColumn

groupContinuedMessage

ungroup

ungroupAll

## **searchPanel**

**highlightCaseSensitive**

Notifies the UI component whether search is case-sensitive to ensure that search results are highlighted correctly. Applies only if highlightSearchText is true. – false

**highlightSearchText**

Specifies whether found substrings should be highlighted.

**placeholder**

Specifies a placeholder for the search panel.

**searchVisibleColumnsOnly**

Specifies whether the UI component should search against all columns or only visible ones.

**text**

Sets a search string for the search panel.

**visible**

Specifies whether the search panel is visible or not.

**width**

Specifies the width of the search panel in pixels.

**sortOrder: undefined | 'asc' | 'desc'**

## sorting**:**

**ascendingText: 'Sort Ascending'**

clearText: Clear Sorting'

descendingText: Sort Descending'

mode: ‘'multiple' | 'none' | 'single'’

**showSortIndexes: true**

## selection

allowSelectAll: true

mode: multiple' | 'none' | 'single'

selectAllMode: allPages' | 'page'

showCheckBoxesMode: 'always' | 'none' | 'onClick' | 'onLongTap'

## stateStoring

State storing enables the UI component to save applied settings and restore them the next time the UI component is loaded. Assign true to the stateStoring.enabled property to enable this functionality.

|  |  |  |
| --- | --- | --- |
| * [filterValue](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/#filterValue) * [focusedRowKey](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/#focusedRowKey) * [selectedRowKeys](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/#selectedRowKeys) * [selectionFilter](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/#selectionFilter) * **filterPanel**.[filterEnabled](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/filterPanel/#filterEnabled) * **paging**.[pageSize](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/paging/#pageSize) * **paging**.[pageIndex](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/paging/#pageIndex) * **searchPanel**.[text](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/searchPanel/#text) | * **columns**.[dataField](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#dataField) * **columns**.[dataType](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#dataType) * **columns**.[filterType](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#filterType) * **columns**.[filterValue](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#filterValue) * **columns**.[filterValues](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#filterValues) * **columns**.[fixed](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#fixed) * **columns**.[fixedPosition](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#fixedPosition) * **columns**.[groupIndex](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#groupIndex) | * **columns**.[name](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#name) * **columns**.[selectedFilterOperation](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#selectedFilterOperation) * **columns**.[sortIndex](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#sortIndex) * **columns**.[sortOrder](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#sortOrder) * **columns**.[visible](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#visible) (only if the [column chooser](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columnChooser/) is [enabled](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columnChooser/#enabled)) * **columns**.[visibleIndex](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#visibleIndex) * **columns**.[width](https://js.devexpress.com/Documentation/20_2/ApiReference/UI_Components/dxDataGrid/Configuration/columns/#width) |

**enabled**: false,

**stateStoring**: {

enabled: true,

type: 'localStorage',

storageKey: 'storage',

}

**savingTimeout**

Specifies the delay in milliseconds between when a user makes a change and when this change is saved. – 2000

**storageKey: keyid for storage**

## **masterDetail:**

enabled: true,

template(),

autoExpandAll – false

## summary: {

recalculateWhileEditing: true,

groupItems: [{

column: 'State',

summaryType: 'count', // 'avg' | 'count' | 'custom' | 'max' | 'min' | 'sum'

displayFormat: '{0} Companies in this state',

alignByColumn: true,

skipEmptyValues: true,

showInGroupFooter: false,

showInColumn: false,

}]}

# **Navigate**

# **Menu:**

adaptivityEnabled:

Specifies whether adaptive UI component rendering is enabled on small screens. Applies only if the orientation is "horizontal". – false

## animation:

hide: { type: 'fade', from: 0, to: 1, duration: 100 }

show: { type: 'fade', from: 0, to: 1, duration: 100 }

dataSource

disabled

disabledExpr

displayExpr

hideSubmenuOnMouseLeave

## items:

beginGroup

closeMenuOnClick

disabled

icon

items

selectable

selected

template

itemsExpr

# **TreeView**

LoadPanel